Laporan Kecerdasan Buatan

Ujian Tengah Semester

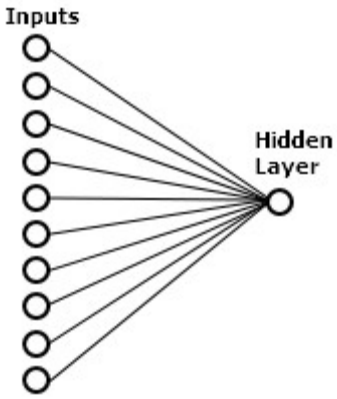
2022

Oleh :
Akhmad Ilham Muharram
21091397009
2021A

Manajemen Informatika
Fakultas Vokasi

1. UTS AI Part 1

A. Single Neuron



Langkah pertama adalah memasukkan perintah import numpy agar mengimport library numpy dan diberi inisial np untuk memudahkan saat digunakan

```
import numpy as np
```

Lalu masukkan perintah untuk membuat variabel inputs yang didalamnya terdapat 10 fitur lapisan

```
inputs = [2,4,1,6,7,2,3,5,9,11]
```

selanjutnya masukkan perintah untuk membuat variabel weights yang didalamnya terdapat Neuron berjumlah 10

```
weights = [  # Neuron
            0.8,0.12,0.5,0.13,0.4,0.2,1,0.6,0.25,0.4
           ]
```

Selanjutnya masukkan perintah untuk membuat variabel bias

```
bias = 3
```

Selanjutnya buat variabel output yang didalamnya terdapat perintah untuk menghitung nya dot product berfungsi sebagai perhitungan sederhana untuk mengalikan dua vector antara variabel inputs dan weights. Penambahan bias dapat dianggap sebagai seberapa fleksibel perceptron tersebut, mirip dengan konstanta b dari fungsi linear y= ax+b memungkinkan untuk memindahkan barisan ke atas dan ke bawah agar sesuai dengan prediksi dengan data yang lebih baik. Perhitungan nya :

$$Output = [(inputs[0] \times inputs[0]) + (inputs[1] \times inputs[1]) + (inputs[2] \times inputs[2]) + (inputs[3] \times inputs[3]) + (inputs[4] \times inputs[4]) + (inputs[5] \times inputs[5]) + (inputs[6] \times inputs[6]) + (inputs[7] \times inputs[7]) + (inputs[8] \times inputs[8]) + (inputs[6] \times inputs[6]) + \text{bias}]$$

```
output = np.dot(weights, inputs) + bias
```

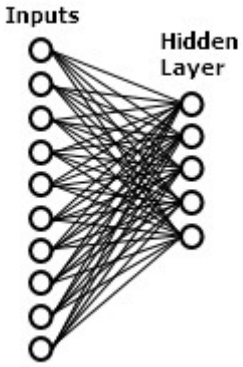
Lalu yang terkhir tulis perintah print output agar bisa di jalankan

```
print(output)
```

Hasil Outputnya

```
22.21
```

B. Multi Neuron



Langkah pertama adalah memasukkan perintah import numpy agar mengimport library numpy dan diberi inisial np untuk memudahkan saat digunakan

```
import numpy as np
```

Lalu masukkan perintah untuk membuat variabel inputs yang didalamnya terdapat 10 fitur lapisan

```
inputs = [2.0,-1.4,3.9,-2.15,4.-0,7.1,5.2,6.4,7.5,-1.12]
```

selanjutnya masukkan perintah untuk membuat variabel weights (matriks) yang didalamnya terdapat 6 Neuron yang masing masing berjumlah 10 angka

```
weights = [
            #Neuron 1
            [0.4,-0.8,0.1,0.2,0.5,-2.0,0.7,0.13,-0.45,0.6],
            #Neuron 2
            [0.26,0.91,0.14,-0.25,0.54,0.52,0.21,-0.82,0.31,-0.42],
            #Neuron 3
            [-0.51,0.44,0.17,-0.32,0.46,0.67,-0.12,0.19,0.34,-0.71],
            #Neuron 4
            [0.2,-0.41,0.3,-0.6,0.52,0.72,0.35,0.66,-0.82,0.12],
            #Neuron 5
            [0.14,0.8,0.75,-0.86,0.41,0-.65,0.22,0.6,0.7,-0.55]]
```

Lalu masukkan variabel biases yang berjumlah sesuai dengan banyaknya Neuron

```
biases = [2.0,4.0,1.5,3.0,0.5]
```

Selanjutnya buat variabel output yang didalamnya terdapat perintah untuk menghitung nya. dot product berfungsi sebagai perhitungan sederhana untuk mengalikan dua vector antara variabel inputs dan weights. Penambahan bias dapat dianggap sebagai seberapa fleksibel perceptron tersebut, mirip dengan konstanta b dari fungsi linear y= ax+b, memungkinkan untuk

memindahkan barisan ke atas dan ke bawah agar sesuai dengan prediksi dengan data yang lebih baik.

Didalam operasi weights digunakan terlebih dahulu daripada inputs karena input didefinisikan array 10 x 1 oleh pyton dan weights 5 x 10 sehingga bisa dikalikan menjadi 5 x 10 . 10 x 1

```
layer_outputs = np.dot(weights, inputs) + biases
```

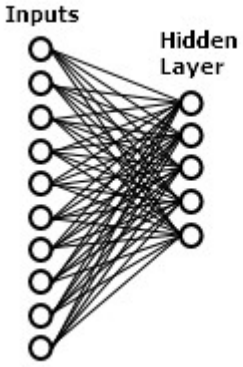
Lalu yang terkhir tulis perintah print output agar bisa di jalankan

```
print(layer_outputs)
```

Hasil Outputnya

```
[-7.895   8.8209 11.7492 13.3856 12.309 ]
```

C. Multi Neuron Batch Input



Langkah pertama adalah memasukkan perintah import numpy agar mengimport library numpy dan diberi inisial np untuk memudahkan saat digunakan

```
import numpy as np
```

Lalu masukkan perintah untuk membuat variabel inputs (matriks) yang didalamnya terdapat 6 input yang masing masing input ada 10 lapisan

```
inputs =[
            # Inputs 1
            [2.0,-1.4,3.9,-2.15,4.0,7.1,5.2,6.4,7.5,-1.12],
            # Inputs 2
            [3.0,1.0,-1.1,3.1,5.1,4.12,3.3,4.0,5.5,-0.5],
            # Inputs 3
            [2.2,-1.14,2.1,0.9,0.45,5.12,-0.11,0.88,-3.55,3.1],
            # Inputs 4
            [6.1,-2.13,2.14,-5.1,6.2,1.1,2.09,0.7,-0.14,0.6],
            # Inputs 5
            [4.1,0.51,0.42,0.42,-0.52,0.31,0.31,-0.43,4.21,3.22],
            # Inputs 6
            [3.33,-4.13,1.53,2.1,5.1,6.12,2.41,-4.32,6.45,0.16]
        ]
```

selanjutnya masukkan perintah untuk membuat variabel weights (matriks) yang didalamnya

terdapat 6 Neuron yang masing masing berjumlah 10 angka

```
weights =[
            # Neuron 1
            [0.4,-0.8,0.1,0.2,0.5,2.0,0.7,-0.13,0.45,0.6],
            # Neuron 2
            [0.26,0.91,0.14,0.25,0.54,-0.52,0.21,0.82,0.31,0.42],
            # Neuron 3
            [0.51,0.44,-0.17,0.32,0.46,0.67,0.12,0.19,-0.34,0.71],
            # Neuron 4
            [0.2,0.41,0.3,-0.6,0.52,0.72,-0.35,0.66,-0.82,0.12],
            # Neuron 5
            [-0.14,0.8,-0.75,0.86,0.41,0.65,0.22,-0.6,0.7,0.55]
        ]
```

Lalu masukkan variabel biases yang berjumlah sesuai dengan banyaknya Neuron

```
biases = [2.0,4.0,1.5,3.0,0.5]
```

Selanjutnya buat variabel output yang didalamnya terdapat perintah untuk menghitung nya. dot product berfungsi sebagai perhitungan sederhana untuk mengalikan dua vector antara variabel inputs dan weights. Penambahan bias dapat dianggap sebagai seberapa fleksibel perceptron tersebut, mirip dengan konstanta b dari fungsi linear y= ax+b, memungkinkan untuk memindahkan barisan ke atas dan ke bawah agar sesuai dengan prediksi dengan data yang lebih baik.

Di dalam operasi terdapat np.array(weights).T yang di definisikan weights array 5 x 10 lalu di transposekan menjadi 10 x 5 supaya bisa dikalikan antara inputs dan array menjadi 6 x 10 . 10 x 5

```
layer_outputs = np.dot(inputs, np.array(weights).T) + biases
```

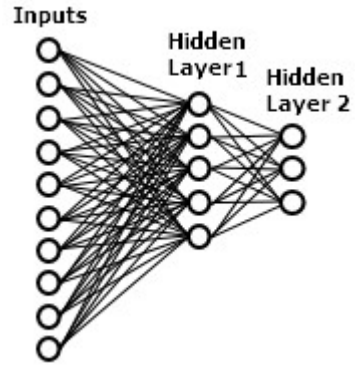
Lalu yang terkhir tulis perintah print output agar bisa di jalankan

```
print(layer_outputs)
```

Hasil Outputnya

```
[[25.591   9.9171  5.6448  8.5976  2.519 ]
 [17.665  12.3906  8.6864  4.3534 11.041 ]
 [14.7181  2.5342  9.2508 10.8853  0.6593]
 [12.307   6.6698  6.1244 10.982  -4.5202]
 [ 7.8174  7.6219  4.6572  0.3978  5.4127]
 [27.2461  0.4487  5.6284 -0.7344 11.1825]]
```

2. UTS AI Part 2



Langkah pertama adalah memasukkan perintah import numpy agar mengimport library numpy dan diberi inisial np untuk memudahkan saat digunakan

```
import numpy as np
```

Lalu masukkan perintah untuk membuat variabel inputs yang didalamnya terdapat 6 input yang masing masing input ada 10 lapisan

```
inputs =[
            # Inputs 1
            [2.0,-1.4,3.9,-2.15,4.0,7.1,5.2,6.4,7.5,-1.12],
            # Inputs 2
            [3.0,1.0,-1.1,3.1,5.1,4.12,3.3,4.0,5.5,-0.5],
            # Inputs 3
            [2.2,-1.14,2.1,0.9,0.45,5.12,-0.11,0.88,-3.55,3.1],
            # Inputs 4
            [6.1,-2.13,2.14,-5.1,6.2,1.1,2.09,0.7,-0.14,0.6],
            # Inputs 5
            [4.1,0.51,0.42,0.42,-0.52,0.31,0.31,-0.43,4.21,3.22],
            # Inputs 6
            [3.33,-4.13,1.53,2.1,5.1,6.12,2.41,-4.32,6.45,0.16]
        ]
```

Inisialisasi variable weights layer pertama yang didalamnya terdapat 5 neuron yang masing masing neuron ada 10 lapisan

```
weights_1 =[
            # Neuron 1
            [0.4,-0.8,0.1,0.2,0.5,2.0,0.7,-0.13,0.45,0.6],
            # Neuron 2
            [0.26,0.91,0.14,0.25,0.54,-0.52,0.21,0.82,0.31,0.42],
            # Neuron 3
            [0.51,0.44,-0.17,0.32,0.46,0.67,0.12,0.19,-0.34,0.71],
            # Neuron 4
            [0.2,0.41,0.3,-0.6,0.52,0.72,-0.35,0.66,-0.82,0.12],
            # Neuron 5
            [-0.14,0.8,-0.75,0.86,0.41,0.65,0.22,-0.6,0.7,0.55]
        ]
```

Inisialisasi variable biases pada layer pertama yang berjumlah sesuai dengan banyaknya Neuron di weights 1 yakni 5

```
biases_1 = [2.0,4.0,1.5,3.0,0.5]
```

Inisialisasi variable weights layer kedua yang didalamnya terdapat 3 neuron yang masing masing neuron ada 5 lapisan dari banyaknya bias pada layer pertama

```
Weights_2 =[
            # Neuron 1
            [1.2,2.2,1.3,2.1,0.2],
            # Neuron 2
            [1.4,2.5,0.6,2.4,0.5],
            # Neuron3
            [2.7,1.8,0.9,2.4,2.1]
        ]
```

Inisialisasi variable biases pada layer kedua yang berjumlah sesuai dengan banyaknya Neuron di weights 2 yakni 3

```
biases_2 = [2.1,1.2,2.3]
```

Selanjutnya buat variabel output pada layer pertama yang didalamnya terdapat perintah untuk menghitung nya. dot product berfungsi sebagai perhitungan sederhana untuk mengalikan dua vector antara variabel inputs dan weights. Penambahan bias dapat dianggap sebagai seberapa fleksibel perceptron tersebut, mirip dengan konstanta b dari fungsi linear y= ax+b, memungkinkan untuk memindahkan barisan ke atas dan ke bawah agar sesuai dengan prediksi dengan data yang lebih baik.

Di dalam operasi terdapat np.array(weights).T yang di definisikan weights 1 array (5 x 10) lalu di transposekan menjadi (10 x 5) supaya bisa dikalikan antara inputs dan array menjadi (6 x 10) . (10 x 5) lalu di jumlahkan dengan biases pertama

```
layer_outputs_1 = np.dot(inputs, np.array(weights_1).T) + biases_1
```

Selanjutnya buat variabel output untuk layer kedua yang didalamnya sama dengan perhitungan untuk output layer pertama bedanya Variabel Inputs pada layer pertama diubah menjadi hasil dari perhitungan layer _output _1 dan variable biases_1 diubah dengan variable biases_2

```
layer_outputs_2 = np.dot(layer_outputs_1,np.array(Weights_2).T)+ biases_2
```

Lalu print layer outputs 2 untuk menampilkan hasil dari perhitungan output layer 2

```
print(layer_outputs_2)
```